DOCUMENTS WIZARD – USER MANUAL

Mateusz Bender

2012-08-07 13:07:00

Project	Documents Wizard
Subject	Documents Wizard User Manual
Author	Mateusz Bender

1.System purpose....1
2.System usage....1
2.1. const tag....1
2.2. contextTree tag....2
2.3. link tag....2
2.4. insert tag....3
2.4.1.Inserting images....3
2.5. ifdef and ifndef tags....3
2.5.1. ifdef.Text and ifndef.Text tags....4
2.6. foreach tag....5
2.6.1. foreach.Text tag....6
2.6.2. foreach.Slide tag....6
2.7.Text formatting....6
2.8.Element types....6
1.System purpose....1
2.System usage....1
2.1. const tag....1
2.2. contextTree tag....2
2.3. link tag....2
2.4. insert tag....3
2.4.1.Inserting images....3
2.5. ifdef and ifndef tags....3
2.5.1. ifdef.Text and ifndef.Text tags....4
2.6. foreach tag....5
2.6.1. foreach.Text tag....6
2.6.2. foreach.Slide tag....6
2.7.Text formatting....6
2.8.Element types....6

1. System purpose

Documents Wizard is used to automatically generate documents based on document templates and CRM data.

2. System usage

To use the system it is necessary to create a document template and upload it to MS CRM 2011. This template can be a typical .docx or .pptx file, containing "tags". Documents

Wizard contains a variety of tags, which can be classified as normal and block (which use 2 tags – opening and closing).

- *_const* tag
- *_contextTree* tag
- *_link* tag
- *_insert* tag
- *_ifdef* and *_ifndef* tags (conditional instructions)
- *_foreach* tag (loop)

Each tag has its own format which must be used. To use a tag in a template it must be entered in the document template in the correct format.

2.1. *_const tag*

The *_const* tag is used to insert predefined, constant fields.

Tag format:

{_const.field}

where:

- *_const* – tag ID
- *field* – field, which will be inserted.

Currently available fields:

- *date* – current date in YYYY-MM-DD format
- *time* – current time in HH:MM format
- *context* – the name of the current context
- *fcontext* – return the full path to the current context (which includes any context nodes "above" the current context)
- *fields* – returns a list of all available fields in the current context.

2.2. *_contextTree tag*

The *_contextTree* tag is used to generate a multi-level list which describes all available fields and context nodes contained within the supplied CRM data. This list will replace the entire paragraph in which the *_contextTree* tag has been used.

Tag format:

{_contextTree}

where:

- *_contextTree* – tag ID.

2.3. *_link tag*

The *_link* tag is used to insert hyperlinks (clickable web links) inside the generated document.

Tag form:

{_link.address}text{_link}

where:

- *_link* – tag ID
- *address* – URL address to be used
- *text* – text which will be displayed instead of the address
- *{_link}* – closing tag (must be used in the same paragraph as the opening tag).

2.4. *_insert tag*

The *_insert* tag is used to insert the content of fields from the current or parent context.

Tag format:

{_insert.field}

or

{_insert.context.field}

where:

- *_insert* – tag ID
- *context* – parent context name from where the field is to be used (context nodes will be described in detail in the *_foreach* tag section)
- *field* – field name; the contents of this field will replace the *_insert* tag

2.4.1. Inserting images

In case of PowerPoint presentations (PPTX files) it is possible to insert images contained inside context fields. To do so, a normal (possibly blank) image object must be placed, and then an actual image source must be defined. Defining a source means changing the "Alternate text" property of the image to an *_insert* tag, using the correct field name (ex. *{_isert.photo}*).

Note!
The generated image will have the same dimensions as the image object that was used to place the *_isert* tag. Make sure to use images with similar proportions as their placeholder images.

2.5. *_ifdef and _ifndef tags*

The *_ifdef* (*if defined*) and *_ifndef* (*if not defined*) tags are used as conditional instructions and allow to check whether a field or sub-context exists (*_ifdef*) or doesn't exist (*_ifndef*). If the condition is not met then the content between the opening and closing tags will not be inserted into the generated document.

Tag format:

{_ifdef.element.field}
content
{_endif}

or

{_ifdef.element.context.field}
content
{_endif}

or

{_ifndef.element.field}
content
{_endif}

or

{_ifndef.element.context.field}
content
{_endif}

where:

- *_ifdef*, *_ifndef* – tag ID
- *element* – the type of the opening element; using the proper element makes it possible to exclude, for example, entire table rows from the final document. Valid elements are, for example, paragraphs (*Paragraph*) or tables (*Table*)
- *context* – the context name, where the field is located
- *field* – the name of the field which will be checked for existence
- *content* – the content which will be deleted in case the condition is not met. It must be contained between the opening and closing tags. The content can contain

any typical document elements, including tables, images as well as other Document Wizard tags (such as *_insert*).
- *{_endif}* – closing tag. It must reside in the same document element as the opening tag.

This tag can be nested, which means that one *_ifdef* tag can contain another *_ifdef* tag. The opening and closing tags must be contained in different paragraphs.

2.5.1. _ifdef.Text and _ifndef.Text tags

These tags are a special case of the *_ifdef* and *_ifndef* tags. These must be placed inside the same paragraph.

2.6. _foreach tag

The *_foreach* repeats a given document fragment for each context node. It also changes the current context. It can be nested and requires and opening and closing tags. Also, the opening and closing tags must be contained in different paragraphs.

Tag format:

{_foreach.element.context}
content
{_foreach}

gdzie:
- *_foreach* – tag ID
- *element* – the type of the opening element
- *context* – the name of the new context
- *content* – similar to the *_ifdef* or _ifndef tags, except the content will not be deleted, but repeated for every context node.
- *{_foreach}* – closing tag.

To describe how context nodes work, it's best to use an example. Each context represents an entity. It contains properties of that entity (contained inside fields) as well as child context nodes. For example, a city could be the top-most context. It contains certain fields (such as a name and area). A city is made out of houses, parks and factories. Each house, park or factory represents a different context. A house can contains further context nodes, such as apartments or offices.

GLOBAL
CITY (1) – name, area
HOUSE (1) – street, number
APPARTMENT (1) – number
APPARTMENT (2)
OFFICE (1)
HOUSE (2)
HOUSE (3)
PARK (1)
FACTORY (1)
CITY (2)

The top-most context is always a GLOBAL element. It contains certain global fields, such as the CRM organization name or the current CRM user. In the above example, inserting a *_foreach* tag for the CITY context will cause whatever content is between the opening and closing tags to be repeated 2 times (2 cities); the first copy will use CITY (1) as the context, while the second copy will use CITY (2). If another *_foreach* tag will be placed inside for the HOUSE context, then whatever is inside those tags will be repeated for every HOUSE in every context, in this case 3 times for CITY (1).

It is therefore possible to create a different table for each CITY, containing rows with properties for every HOUSE.
It is possible to reference fields in a parent context. In the above example, if the current context is HOUSE (1), it is possible to insert the name of CITY (1) using the *_insert* tag:

{_insert.CITY.name}

2.6.1. _foreach.Text tag

This tag has the same function as the normal *_foreach* tag, however the opening and closing tags must be inside the same paragraph. It allows, for example, entering values separated by a comma.

2.6.2. _foreach.Slide tag

This tag is used only in PowerPoint (PPTX) documents. The opening and closing tags must be contained in separate paragraphs, however, they can be contained in different slides (unlike any other tag used in PPTX documents). It doesn't matter where the tag is placed; Documents Wizard will repeat the entire slide where the tag was used and remove the tag itself from the final document.

2.7. Text formatting

Applying text formatting to the tags (such as fonts, font sizes, bold or underline, etc.) causes the entered text to use that formatting.

2.8. Element types

Below are some of the most commonly used element types in DOCX files:

- *Paragraph* – represents a single paragraph
- *TableRow* – represents a table row
- *Table* – represents and entire table
- *TableCell* – represents a single table cell